Compact and Low-complexity Binary Feature Descriptor and Fisher Vectors for Video Analytics

Roberto Leyva, Victor Sanchez, *Member IEEE*, and Chang-Tsun Li, *Senior Member IEEE*

***Abstract*—In this paper, we propose a compact and low-complexity binary feature descriptor for video analytics. Our binary descriptor encodes the motion information of a spatio-temporal support region into a low-dimensional binary string. The descriptor is based on a binning strategy and a construction that binarizes separately the horizontal and vertical motion components of the spatio-temporal support region. We pair our descriptor with a novel Fisher Vector (FV) scheme for binary data to project a set of binary features into a fixed length vector in order to evaluate the similarity between feature sets. We test the effectiveness of our binary feature descriptor with FVs for action recognition, which is one of the most challenging tasks in computer vision, as well as gait recognition and animal behavior clustering. Several experiments on the KTH, UCF50, UCF101, CASIA-B, and TIGdog datasets show that the proposed binary feature descriptor outperforms the state-of-the-art feature descriptors in terms of computational time and memory and storage requirements. When paired with FVs, the proposed feature descriptor attains a very competitive performance, outperforming several state-of-the-art feature descriptors and some methods based on convolutional neural networks.**

***Index Terms*—binary features, video analysis, Fisher Vectors, CNN.**

I. Introduction

Nearly a million minutes of video content circulate on the internet every second [1]. Due to this dramatic growth, it is necessary to explore and design efficient methods to analyze video data [2], e.g., for summarization [3], classification [4, 5], action recognition [6, 7], video surveillance [8, 9], and abnormal event detection [10, 11]. Convolutional Neural Networks (CNNs) and approaches based on hand-crafted feature descriptors have demonstrated outstanding performance for several video analysis tasks. However, in many cases, these approaches may require a vast amount of computational resources to extract and process the extracted features. This is a major concern for CNNs, which may require up to months to complete the training process even if clusters or powerful servers are used [12]. Although methods based on hand-crafted feature descriptors tend to require shorter training and processing times than those based on deep neural networks, the high-dimensionality of some of the extracted feature descriptors demands vast storage capacities and computational resources, thus hindering their implementation on low-cost devices and conventional machines [7, 13]. Developing low-complexity, compact and efficient feature descriptors for video analytics is therefore an important challenge.

Inspired by other efficient and low-complexity binary feature descriptors for images [14–21], this work further improves our binary feature descriptor for video in [22, 23] by increasing its descriptiveness power and reducing its computational complexity. Furthermore, we propose to pair our descriptor with Fisher Vectors (FVs) for binary data, which is a high-order mapping technique that projects a set of binary features into a fixed-length vector to effectively evaluate the similarity among feature sets. The main contributions of this work are as follows:

- Our binary feature descriptor captures motion information from two sources, i.e., optical flow and temporal gradients. The latter is encoded independently for two motion components, which increases the descriptiveness power.
- We introduce a novel technique based on Integral Video (IV) to considerably reduce the computational complexity of our binary feature descriptor.
- We propose a novel FV formulation based on the Perronin's [24] assumptions of the Fisher kernel built on Bernoulli distributions, which allows simplifying the Information Matrix. To the best of our knowledge, this work is the first to formulate FVs using such assumptions on binary data.
- We paired our binary feature descriptor with FVs to project a set of binary features into a fixed-length vector. To the best of our knowledge, this work is the first to apply FVs to binary feature vectors extracted from video data.

To evaluate the compactness, computational complexity and descriptiveness power of our binary feature descriptor, we focus on action recognition, which is one of the most challenging tasks in computer vision, as well as gait recognition and animal behavior clustering. Evaluations on the KTH, UFC101, UCF50, CASIA-B and TIGdog datasets show that our feature descriptor, when paired with our FVs, attains a very competitive performance with short processing times and low storage requirements, outperforming several binary and non-binary feature descriptors, as well as some methods based on CNNs.

The rest of the paper is organized as follows. Section II presents a brief review of existing state-of-the-art feature descriptors and high-order mapping techniques for video analytics. This sections highlights the efforts that have been made

Roberto Leyva, Victor Sanchez, and Chang-Tsun Li are with the Computer Science Department at University of Warwick, Coventry, United Kingdom.

Roberto Leyva is also with the University of Essex. Chang-Tsun Li is also with the School of Information Technology, Deakin University, Geelong, Australia.

This work was funded by the Mexican Ministry of Education - CONACyT and the EU Horizon 2020 project, entitled Computer Vision Enabled Multimedia Forensics and People Identification (acronym: IDENTITY, Project ID: 690907)

to reduce the computational complexity and dimensionality of feature descriptors. In Section V, we explain in detail our improved binary feature descriptor and FV formulation. This section also includes a discussion of the complexity of our binary feature descriptor compared to that of CNNs. Experimental results and discussions are presented in Section V. We conclude this paper in Section VI.

II. Previous Work

Local features: Features extracted from Spatio-Temporal Support Regions (STSR), i.e., local features [25–34], have been shown to attain a remarkable performance in various tasks including classification/recognition [6, 35, 36], summarization [3], and anomaly detection [11]. These local features are usually extracted from various motion information sources, such as intensity [25, 26], frequency transformations [27–29], 3D gradients [30–32], voxel texture [36] and optical flow [33, 34]. The descriptive power of their associated feature vectors [26, 37] is further improved by tracking and coding information from inhomogeneous patches [33, 34, 38–41], which is especially useful in complex backgrounds [42]. Unfortunately, besides their high-dimensionality [31–33], representing numerically these local features requires double-precision numbers [30–33, 33, 34]. Consequently, any further processing of the feature vectors, e.g., clustering and matching, inevitably involves very long computational times [19, 21, 31] and large memory and storage requirements [33, 34]. Important efforts have been made to tackle the high-dimensionality and high-complexity of local features [14, 43–47]. These efforts mainly include double-precision to binary projections [44–46, 48], which may be applied not only to local features [49], but also to those features extracted globally as high-order representations [43, 47]. Binary feature descriptors for video that do not rely on double-precision representations, e.g., [44, 45], have been rarely explored because of their relatively low-performance [50]. For the case of images, however, binary feature descriptors have been shown to attain a very good performance [17–21]. This motivates us to design a low-complexity and compact binary feature descriptor for video.

Fisher Vectors for binary data: Local features combined with FVs [34, 51, 52] have attracted much attention because of their superior performance compared to Bag of Features (BoF) representations [53]. FVs provide a more general way to define a kernel from a generative process of the data, as compared to BoF representations [24]. It is important to note that BoFs are a particular case of FVs, in which the gradient computation is restricted to the weight parameters of the mixture defining the probabilistic projection model. This inevitably hinders its accuracy for various classification tasks [53]. Because FVs can represent higher order information than BoFs [24, 52], they can outperform these representations [34, 35, 52, 54]. For the specific case of binary feature descriptors, FVs have been successfully employed in image retrieval tasks [55]. This motivates us to pair our binary feature descriptor with FVs.

III. Proposed Spatio-Temporal Binarization

Our binary feature descriptor, hereafter called Spatio-Temporal Binarization (STB), encodes the motion information of a video volume as a concatenation of three binary strings, as depicted in Fig. 1. The encoded motion information is obtained from two motion sources: optical flow and temporal gradients, which have been shown to provide rich motion information by considering pixel intensity changes to create a new data space that disregards the background [25]. The first binary string of STB represents the video volume's motion information extracted from optical flow (see orange block – Binary Trajectory Encoding (BiTE) in Fig. 1). The second and third binary strings of STB represent the horizontal and vertical motion components of the video volume's temporal gradients (see blue block – Binary Integral Video Encoding (BIVE) in Fig. 1). We detail next the computation of the binary strings comprising STB.

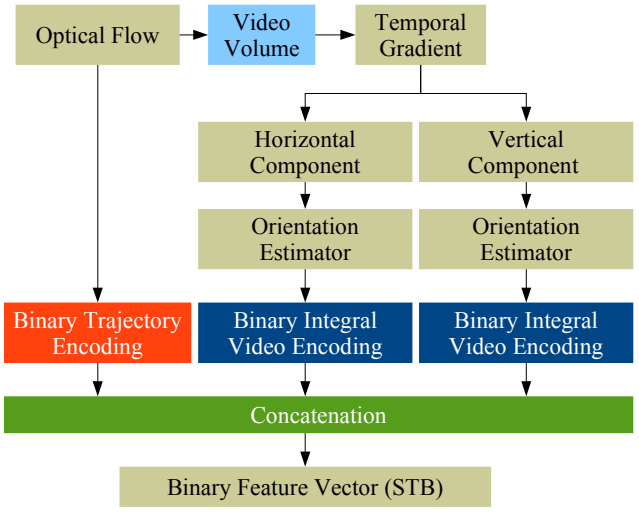


Fig. 1: STB descriptor. Each STSR, or video volume, is encoded as a concatenation of three binary strings: 1) BiTE encodes the video volume's motion information extracted from optical flow. BIVE separately encodes the 2) horizontal and 3) vertical motion components of the temporal gradients.

A. Binary Trajectory Encoding - BiTE

BiTE encodes the trajectories described by points tracked after computing the optical flow of the video data, e.g., by using the improved dense trajectory detector [34, 41]. The trajectory of each tracked point comprises L displacement vectors representing motion information in time. Each trajectory, T_i, defines a STSR, or video volume, v_i, of size $n_x \times n_y \times n_t$ as Fig. 2.a illustrates. BiTE first normalizes T_i with respect to its largest displacement vector to produce $\hat{T}_i$:

$$\hat{T}_i = \frac{(\Delta p_t, ..., \Delta p_{t+L-1})}{\max \|\Delta p\|}, \quad (1)$$

where p_t is the point tracked at time t, and Δp_t is the corresponding displacement vector. BiTE then employs an orientation-invariant binning strategy that maps the normalized displacement vectors, $\Delta \hat{p}_t$, into one of $Q = 6$ 4-bit binary

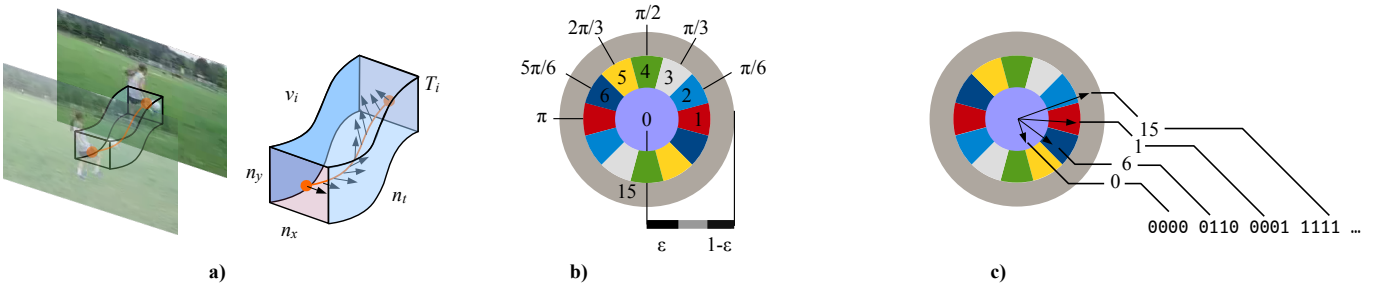


Fig. 2: Binary Trajectory Encoding. **a)** A video volume, i.e., a STSR, is defined for each trajectory, which comprises L displacement vectors. **b)** The direction and magnitude of the constituent displacement vectors are encoded using a binning stratregy. **c)** Example binning of 4 normalized displacement vectors.

codes, each representing a bin:

$$B(\Delta\hat{p}_t) = \begin{cases} q : \arg\min_q \parallel \Delta\hat{p}_t - b_q \parallel & : |\Delta\hat{p}_t| \in (\epsilon, 1-\epsilon) \\ 15_{\texttt{dec}} & : |\Delta\hat{p}_t| \geq 1-\epsilon \\ 0_{\texttt{dec}} & : |\Delta\hat{p}_t| \leq \epsilon \end{cases}, \tag{2}$$

where b_q is a unitary vector that represents the main direction associated with the qth bin. It is important to note that the binning strategy in Eq. 2 not only considers the direction of each displacement vector, but also their normalized magnitude. The latter is important to improve the descriptiveness power of the resulting binary codes because very small or very large displacements vectors tend to have unstable directions. Specifically, very small displacement vectors are prone to have random directions, while very large displacement vectors are likely to have been generated by camera motion. Eq. 2 then bins $\Delta\hat{p}_t$ into one of $Q = 6$ bins *if and only if* its magnitude is within the range $[\epsilon, 1-\epsilon]$, which is the range of normalized magnitudes that captures the most stable directions. Those displacement vectors with normalized magnitudes $\leq \epsilon$ are represented by the binary code `0000`, or 0_{dec}, while those with normalized magnitudes $\geq 1-\epsilon$ are represented by the binary code `1111`, or 15_{dec}.

Table I tabulates the details of the $Q = 6$ bins used to encode the displacement vectors with normalized magnitudes $\in [\epsilon, 1-\epsilon]$, i.e., the stable displacement vectors. Note that compared with our previous work in [23], which employs shorter binary codes, by using 4-bit codes we can uniquely represent the six bins and use the most significant bit (MSB) to distinguish those very large, unstable displacement vectors, whose MSB is always 1. This increases the power of the descriptor to distinguish between stable and unstable displacement vectors. In order to make the encoding direction-invariant, the qth bin has a period of π. Fig. 2.b illustrates this concept, where the bins depicted in the same color are associated with the same binary string. Fig. 2.c illustrates the binning of 4 example normalized displacement vectors.

The binary vector, F_i, representing trajectory $\hat{T}_i$ is generated by concatenating the L binarized displacement vectors:

$$F_i = \sum_{0 \leqslant t < L} 2^{4t} B(\Delta\hat{p}_t). \tag{3}$$

TABLE I: Bins used to encode the direction of stable displacement vectors.

Bin index (q)	Main direction	Binary string	Range of directions
1	0	0001	$\pm\pi/12$
2	$\pi/6$	0010	$\pi/6 \pm \pi/12$
3	$\pi/3$	0011	$\pi/3 \pm \pi/12$
4	$\pi/2$	0100	$\pi/2 \pm \pi/12$
5	$2\pi/3$	0101	$2\pi/3 \pm \pi/12$
6	$5\pi/6$	0110	$5\pi/6 \pm \pi/12$

B. Binary Integral Video Encoding - BIVE

BIVE is based on our previous binary descriptor Binary Wavelet Differences (BWD) [23]. BIVE further improves the descriptiveness power of BWD by separately encoding the vertical and horizontal components of the STSR. Moreover, BIVE reduces the number of computations by using a technique based on IVs. BIVE comprises two main steps: 1) motion component calculation and 2) binary string formation, as Fig. 1 depicts.

1) Motion component calculation: To compute the horizontal and vertical motion components of v_i , denoted by v_i^x and v_i^y, respectively, BIVE first applies a temporal difference operator $d_t = [-1, 1]$ to discard background information. It then extracts v_i^x and v_i^y by applying the Prewit convolution operators $d_x = [-1, 0, 1]$ and $d_y = [-1, 0, 1]^\top$, respectively:

$$v_i^x = |d_x * (d_t * v_i)|, \tag{4a}$$
$$v_i^y = |d_y * (d_t * v_i)|. \tag{4b}$$

In order to make the binary string representing each component orientation-invariant, BIVE rotates v_i^x and v_i^y. To this end, we extend the 2D Rosin operator [56] to the spatio-temporal domain to determine the components' orientations. For each component, the coordinates of the centroid, c, are determined using the center of mass, m:

$$c = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}), \tag{5a}$$
$$m_{pq} = \sum_{1 \leqslant x \leqslant n_x} \sum_{1 \leqslant y \leqslant n_y} \sum_{1 \leqslant t \leqslant n_t/2} x^p y^q v_i^s(x, y, t), \tag{5b}$$

where $s \in \{x, y\}$ denotes the motion component being considered. After setting the origin, o, of the reference at the

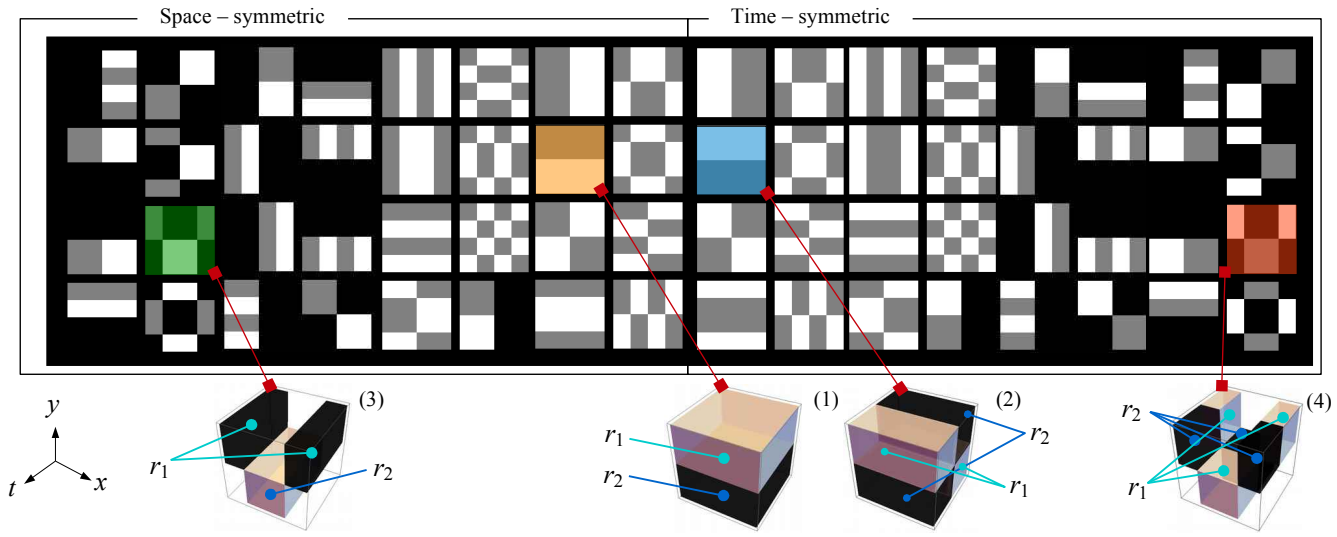


Fig. 3: Wavelet-based patterns. Each pattern defines two regions, r_1 and r_2. Four sample patterns are illustrated: (1) is complementary to (2), while (3) is complementary to (4).

geometric center, $(n_x/2, n_y/2)$, the component's orientation is given by the angle θ of $r = \overline{oc}$:

$$\theta = \texttt{atan2}\,(r_y, r_x)\,. \tag{6}$$

Finally, θ is binned into four quadrant directions, i.e., $\theta \in \{0, \pi/2, \pi, 3\pi/2\}$, and v_i^x and v_i^y are rotated using the flip operation:

$$v_i^s(x, y, t) = \begin{cases} v_i^s(x, n_y - y, t), & \theta \simeq \pi/2 \\ v_i^s(n_x - x, y, t), & \theta \simeq 0 \end{cases}. \tag{7}$$

2) Binary string formation: BIVE separately encodes the horizontal and vertical motion components v_i^x and v_i^y into a binary string. The encoding process is based on comparing the summed values of several pairs of relatively large regions defined within each motion component to create logical values. BIVE uses wavelet-based patterns to define the pairs of regions of v_i^x and v_i^y to be compared [20, 21, 57], which avoids seeking the best pairs, as required when using small isolated regions [18, 21]. Although it is possible to use other patterns, e.g., those used by FREAK [21], ORB [18] and DAISY [20], our evaluations show that other patterns provide inferior results than those provided by our wavelet-based patterns.

Fig. 3 depicts the $K = 64$ patterns used by BIVE. Note that each pattern indeed divides a video volume into two regions, denoted by r_1 and r_2. These patterns are symmetrical in either space or time. For example, patterns (1) and (3) are space-symmetric while patterns (2) and (4) are time-symmetric. The difference between patterns (3)-(4) and (1)-(2) is that the latter pair contains void regions that are not considered during the encoding. Patterns with void regions reduce the overlap of the analysed STSR, which helps to produce very descriptive binary strings. Time-symmetric patterns are computed as the complement to space-symmetric patterns in the range $[0, t/2]$; e.g., (1) and (2), where (2) is complementary to (1). A pattern complementary to a symmetric one is therefore always time-symmetric and generated from a space-symmetric pattern. It is important to note that the inclusion of time-symmetric patterns in this work is an important improvement compared to our previous work in [22, 23]. Time-symmetric and space-symmetric patterns allow fast and slow motion, respectively, to be captured thus improving the descriptiveness of the resulting binary strings compared to those in [23].

BIVE generates a bit by comparing the summed values of the two regions, r_k^1 and r_k^2, generated by the kth pattern. This process is a K-dimensional mapping, $\mathbb{R}^3 \rightarrow \mathbb{R}^K$. The kth bit for regions r_k^1 and r_k^2 is computed as follows:

$$C(r_k^1, r_k^2) = \begin{cases} 1, & \text{if } \sum\left(v\left(r_k^1\right)\right) > \sum\left(v\left(r_k^2\right)\right) \\ 0, & \text{otherwise} \end{cases}, \tag{8}$$

where $\sum\left(v\left(r_k^n\right)\right)$ represents the summed values of region n. BIVE then calculates a binary string, G_i, one for each motion component, v_i^x and v_i^y, as the concatenation of all K comparisons:

$$G_i = \sum_{0 \leqslant k < K} 2^k C(r_k^1, r_k^2). \tag{9}$$

Integral Video (IV): For the different $K = 64$ patterns in Fig. 3, Eq. 8 adds the same values several times, making the computation of G_i highly repetitive. To reduce the number of computations required by Eq. 8, BIVE uses the IV of each motion component as a way to pre-compute the summations.

Let $V_{i,s}^p$ denote the IV of the motion component s of volume v_i with $s \in \{x, y\}$. Algorithm 1 details how to compute $V_{i,s}^p$ with a low complexity of $O(n^3)$ by adding the values of v_i^s up to point p:

$$V_{i,s}^p = \sum_{1 \leq x \leq x_p} \sum_{1 \leq y \leq y_p} \sum_{1 \leq t \leq t_p} v_i^s(x, y, t), \tag{10}$$

where (x, y, t) denote the coordinates of v_i^s. The values of $V_{i,s}^p$ at specific locations are then the summed values required by Eq. 8. For example, Fig. 4.b shows how to compare regions r^1 and r^2 by using the summed values at locations a and b of the

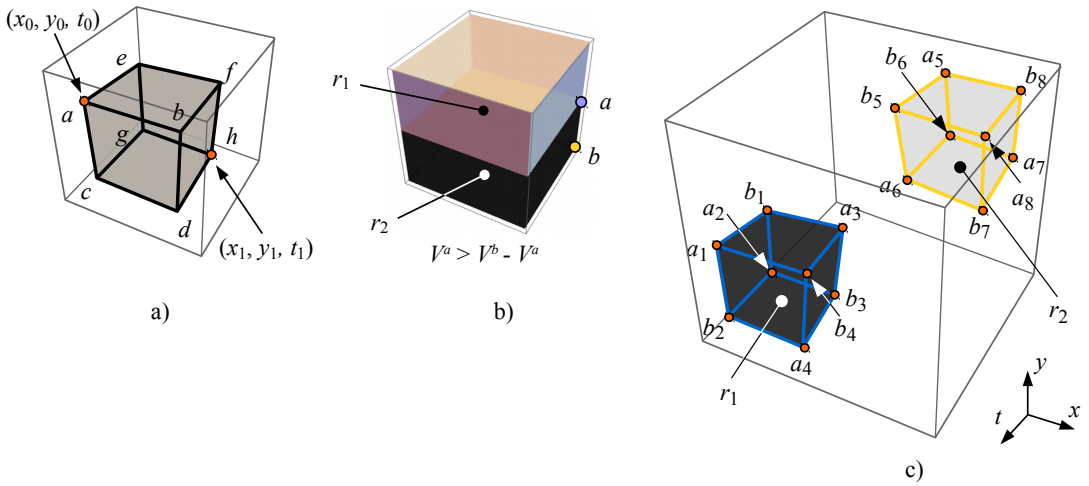


Fig. 4: a) Sub-volume defined by the eight points $a - h$. b) Comparing the summed values of regions r_1 and r_2 requires one subtraction after computing the IV. c) Sets of points, $A = \{a_1, a_2, ..., a_8\}$ and $B = \{b_1, b_2, ..., b_8\}$, defining two regions, r_1 and r_2, to be compared.

corresponding IV, i.e., V^a and V^b, where $\sum\left(v\left(r_k^1\right)\right) = V^a$ and $\sum\left(v\left(r_k^2\right)\right) = (V^a - V^b)$.

Algorithm 1: Integral Video calculation

Input : motion component v_i^s of size (n_x, n_y, n_t), with $s \in \{x, y\}$
Output : summed volume $V_{i,s}^p$ of size (x_p, y_p, t_p)
Variables: acc stores the summation of v_i^s up to location (x, y, t)
v_i^s is also used to store the summed values

```
acc = 0;
for t ∈ (1, t_p) do
    for y ∈ (1, y_p) do
        acc = 0;
        for x ∈ (1, x_p) do
            acc = acc + v_i^s(x, y, t);
            if y > 1 then
                v_i^s(x, y, t) = acc + v_i^s(x, y − 1, t);
                continue
            end
            v_i^s(x, y, t) = acc;
        end
    end
    if t > 1 then
        for y ∈ (1, y_p) do
            for x ∈ (1, x_p) do
                v_i^s(x, y, t) = v_i^s(x, y, t) + v_i^s(x, y, t − 1);
            end
        end
    end
end
V_{i,s}^p = v_i^s(1 : x_p, 1 : y_p, 1 : t_p)
```

Let us denote as $A = \{a_1, a_2, ..., a_8\}$ and $B = \{b_1, b_2, ..., b_8\}$ the sets of points defining the two regions, r_1 and r_2, to be compared, as shown in Fig. 4.c. It can be easily shown that Eq. 8 can be expressed as:

$$C(r_k^1, r_k^2) = \begin{cases} 1, & \text{if } V_k^A > V_k^B \\ 0, & \text{otherwise} \end{cases}, \tag{11}$$

where $V_k^A = V^{a_1} + V^{a_2} + ... + V^{a_8}$ and $V_k^B = V^{b_1} + V^{b_2} + ... + V^{b_8}$.

Eq. 11 has a $O(n)$ complexity. For the $K = 64$ patterns depicted in Fig. 3, the size of A and B depends on the complexity of the regions. For instance, pattern (1) requires fewer points in A and B than pattern (4), and consequently, fewer operations to compute Eq. 11.

After computing the IVs for the two motion components, v_i^x and v_i^y, a minimum of 24 Operations Per Bit (OPB), i.e., additions/subtractions, on average, are required to compute Eq. 11 for the $K = 64$ patterns depicted in Fig. 3, with a maximum of 150 OPB, on average, regardless of the motion component's size. On the contrary, when the IV technique is not used, the number of OPB depends on the motion component's size, i.e., $n_x \times n_y \times n_t$ OPB.

The final STB descriptor for v_i is the binary feature vector, H_i, of dimensions $D = 4L + 2K$, that results from the concatenation of the binary string produced by BiTE (see Eq. 3), and the two binary strings produced by BIVE (Eq. 9), one for each motion component, v_i^x and v_i^y:

$$H_i = F_i +\!\!+ G_i^x +\!\!+ G_i^y. \tag{12}$$

IV. Fisher Vectors for Binary Data

We propose to pair our STB descriptor with FVs. Our FVs are based on the work of Uchida et al. [55]. However, we follow important assumptions introduced in the work of Perronin [51] to derive the Fisher Score and Information Matrix, as explained in Appendices A and B.

Let us denote the binary feature vector, H_i, as computed by Eq. 12, by x_t. Thus $x_t \in \{0, 1\}^D$ is an D-dimensional binary feature. To generate the Bernoulli Mixture Model (BMM), let us define an input vector X as comprising a total of T binary features; i.e., $X = \{x_1, x_2, \ldots, x_T\}$. For an N-component BMM, we define the model's distribution parameters as the set $\theta = \{w_i, \mu_{id} \mid i \in [1, N], d \in [1, D]\}$, where w_i is the weight of the ith BMM component, and μ_{id} is the corresponding mean

across the dth dimension. The probabilistic density function for the T binary features in X is given as:

$$p(X|\theta) = \prod_{1 \leqslant t \leqslant T} p\left(x_t|\theta\right), \tag{13a}$$

$$p(x_t|\theta) = \sum_{1 \leqslant i \leqslant N} w_i p_i\left(x_t|\theta\right), \tag{13b}$$

$$p_i(x_t|\theta) = \prod_{1 \leqslant d \leqslant D} \mu_{id}^{x_{td}} \left(1-\mu_{id}\right)^{1-x_{td}}, \tag{13c}$$

where x_{td} represents the dth bit of x_t. The parameter set θ is estimated using the Expectation Maximization (EM) algorithm [58]. Specifically, the expectation step calculates the posterior probability, $\gamma_t(i) = p\left(i|x_t, \theta\right)$, of feature x_t generated by the ith BMM component as follows:

$$\gamma_t(i) = \frac{w_i p_i\left(x_t|\theta\right)}{\sum\limits_{1 \leqslant j \leqslant N} w_j p_j\left(x_t|\theta\right)}. \tag{14}$$

In the maximization step, the parameters are updated as follows:

$$S_i = \sum_{1 \leqslant t \leqslant T} \gamma_t(i), \;\; w_i = S_i/T, \;\; \mu_{id} = \frac{1}{S_i} \sum_{1 \leqslant t \leqslant T} \gamma_t(i) x_{td}, \tag{15}$$

where S_i is the zero-order statistic. Parameters w_i and μ_{id} are initialized to $1/N$ and a uniform distribution, $U(1/4, 3/4)$, respectively, as suggested in [55].

Once the parameters converge or the EM reaches a maximum number of iterations, we proceed to map the features using the set of parameters, θ. Differently from the work by Uchida et al. [55], in this work we derive the FV from the BMM following the standard gradient derivation proposed in [51] with a Gaussian Mixture Model (GMM). This guarantees that the Information Matrix do not become undefined for small and large values of the distribution's mean, μ_{id}.

A gradient vector describes the direction to which the parameters should be modified to best fit the data, X. Let us describe X by the gradient $\mathcal{G}_\theta^X$, also known as Fisher Score:

$$\mathcal{G}_\theta^X = \frac{1}{T} \nabla_\theta \mathcal{L}\left(X|\theta\right), \tag{16a}$$

$$\mathcal{L}\left(X|\theta\right) = \log p\left(X|\theta\right). \tag{16b}$$

From Eq. 13a, and assuming independence over the BMM components, the Fisher Score can be expressed in terms of the distribution parameter μ_{id} (see Appendix A for derivation):

$$\mathcal{G}_{\mu_{id}}^X = \frac{1}{T} \sum_{1 \leqslant t \leqslant T} \gamma_t(i) \prod_{1 \leqslant d \leqslant D} \frac{x_{td} - \mu_{id}}{\mu_{id}(1-\mu_{id})}. \tag{17}$$

Let us now consider a class of parametric models $P(X|\theta)$, where $\theta \in \Theta$ defines the Riemannian manifold M_θ with a local metric given by the Information Matrix $F = E_X\{\mathcal{G}^X \left(\mathcal{G}^X\right)^\top\}$. The Fisher Score $\mathcal{G}_\theta^X$ maps X into a new feature vector, i.e., $X \rightarrow \mathcal{G}^X$, which is a point in the gradient space of the manifold M_θ. Specifically, the mapping is given by $\mathcal{G}_\theta^X = \nabla_\theta \log p\left(X|\theta\right)$. The natural kernel, κ, of this mapping is the inner product between the Fisher Score relative to the local Riemannian metric of two sets of features, X and Y:

$$\kappa(X, Y) = \mathcal{G}_\theta^X F_\theta^{-1} \mathcal{G}_\theta^Y. \tag{18}$$

The inner product defines an Euclidean metric that implicitly defines a pseudo-metric in the original feature space via a second-degree polynomial expansion of the kernel [59]. Therefore, the natural kernel is a strong similarity measure in the projected space based on the Euclidean distance. In this work, we calculate the Information Matrix required by Eq. 18 in terms of the distribution parameter μ_{id} as follows (see Appendix B for derivation):

$$F_{\mu_{id}} = \frac{T w_i}{\mu_{id} - \mu_{id}^2}. \tag{19}$$

The FV is a two-normalization of the score concatenation $z = F_{\mu_{id}}^{1/2} \mathcal{G}_{\mu_{id}}^X$. We first use power normalization with coefficient $\alpha \in (0, 1)$, as follows:

$$f(z) = \texttt{sign}(z)|z|^\alpha. \tag{20}$$

We then normalize $f(z)$ using ℓ_2 normalization [24] to compute the final FV, i.e., $Z = \| f(z) \|_2$. For an N-component BMM of binary features with D dimensions, the FV has $N \times D$ dimensions as a result of the concatenation of the feature projections onto each individual BMM component.

It is important to mention three main differences between our FV formulation and that proposed by Uchida et al. [55]:

1) When deriving the Fisher Score (Eq. 17) and Information Matrix (Eq. 19), we do not consider the two possible values that the STB descriptor can take (i.e., 0 or 1) to simplify the computations. Instead, we fully expand the expressions and compute the derivative (see Appendices A and B). This results in expressions that are much simpler to evaluate than those in [55]. Specially, the expressions of Uchida et al. [55] are:

$$\mathcal{G}_{\mu_{id}}^X = \frac{1}{T} \sum_{1 \leqslant t \leqslant T} \gamma_t(i) \prod_{1 \leqslant d \leqslant D} \frac{(-1)^{1-x_{td}}}{\mu_{id}^{x_{td}}(1-\mu_{id})^{1-x_{td}}}, \tag{21}$$

and

$$F_{\mu_{id}} = T w_i \left(\frac{\sum_{j=1}^N w_j \mu_{jd}}{\mu_{id}^2} + \frac{\sum_{j=1}^N w_j \mu_{jd}}{\left(1-\mu_{id}\right)^2} \right). \tag{22}$$

2) Our Information Matrix expression (Eq. 19) allows calculating the FVs without overflow. Specifically, when computing the product $z = F_{\mu_{id}}^{1/2} \mathcal{G}_{\mu_{id}}^X$, it is easy to see that the means in the denominators are to be multiplied. Because Eq. 22 generates smaller values than Eq. 19, the former is more prone to overflow for small values of μ_{id}, i.e., when $\mu_{id} \rightarrow 0$. Our proposed Information Matrix is then more robust to this potential problem. In other words, Eq. 22 becomes undefined for small and large values of this mean. On the contrary, our Information Matrix is defined for a larger range of mean values (see Fig. 5).
3) Since μ_{id} is constant, all the denominators for Eq. 17 can be computed as:

$$\beta_{id} = \frac{1}{\mu_{id}(1-\mu_{id})}. \tag{23}$$

The Fisher Score can then be calculated as:

$$\mathcal{G}^{X}_{\mu_{id}} = \frac{1}{T} \sum_{1 \leqslant t \leqslant T} \gamma_t(i) \prod_{1 \leqslant d \leqslant D} \beta_{id} \left(x_{td} - \mu_{id} \right). \quad (24)$$

Compared to Eq. 21, our expression for the Fisher Score can then be computed without divisions, which gives gains in terms of processing speed.

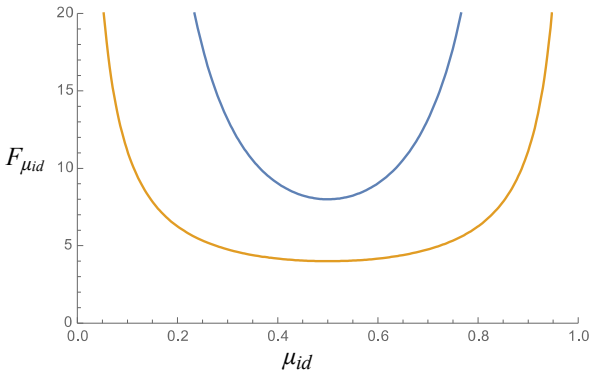


Fig. 5: Information Matrix per individual component as proposed in [55] (blue) and in this work (orange)

V. Experiments and Discussions

Five sets of experiments are conducted. The first set is aimed at confirming the descriptive power and low-dimensionality of BIVE. The second set compares the memory demands and computational time of STB with those of state-of-the-art non-binary and binary feature descriptors. The third set evaluates STB when paired with FVs for action recognition. The fourth set evaluates STB when paired with FVs for gait recognition and animal behavior clustering. The last set of experiments evaluates the effect of the FV parameters on the computational times and action recognition accuracy.

A. First set of experiments: BIVE performance

We evaluate the performance of BIVE against HOF [60], HOG [60], PCA-gradients [61], and several binary feature descriptors. Specifically, we compare it against motion FREAK (moFREAK) [50], the hashing techniques SSH, LSH [44, 45] and IQT [48], and a 3D version of FREAK, BRISK, ORB and BRIEF. The SSH, LSH, and IQT techniques map double-precision feature vectors to binary format. We apply these techniques to HOF features.

We extend FREAK, BRISK, ORB and BRIEF to the spatio-temporal domain by defining the pair of regions to be compared in 3D, as illustrated in Fig. 6. We call these extended 3D feature descriptors 3D-FREAK, 3D-BRISK, 3D-ORB and 3D-BRIEF, respectively. We employ the spatio-temporal detector of Laptev in [37], and a BoF+SVM pipeline. For the KTH dataset, we use the 8/9 split originally proposed by [62]. For the UCF50 dataset, we divide the 50 actions into 25 splits. The average CCR over the splits are reported in Tables II and III.

TABLE II: CCR for action recognition using various feature descriptors on the KTH dataset under a BoF+SVM pipeline

Feature descriptor	CCR	Feature vector dimension
HOF [60] ‡	89.5%	90
SSH-HOF [44] †	87.02%	64
LHS-HOF [45]†	80.55%	72
IQT-HOF [48]†	87.5%	64
moFREAK [50]	82.87%	64
3D-FREAK [21]	(76.38 – 81.94)%	32 – 1024
3D-BRISK [17]	(75.46 – 80.55)%	32 – 1024
3D-ORB [18]	(75.92 – 82.40)%	32 – 1024
3D-BRIEF [19]	(75.46 – 78.70)%	32 – 1024
BIVE	88.88%	2×64

† Binary mapping of double-precision feature vectors.
‡ Double-precision feature vectors.

TABLE III: CCR for action recognition using various feature descriptors on the UCF50 under a BoF+SVM pipeline.

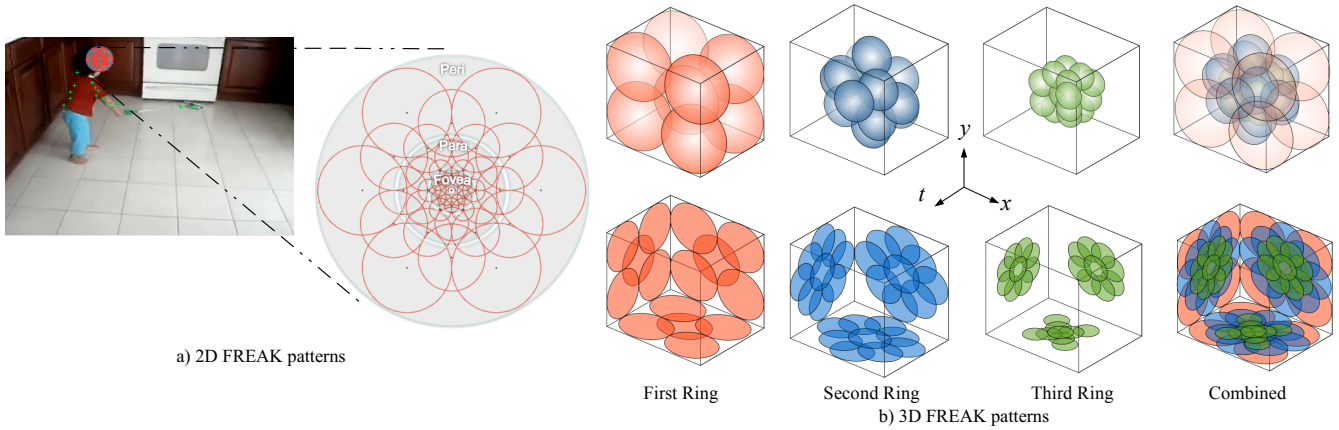
Feature descriptor	CCR	Feature vector dimension
HOF [60] ‡	55.55%	90
HOG [60] ‡	52.45%	72
PCA-Gradients [61]‡	53.06	-
3D-FREAK [21]	42.76%	64
3D-BRISK [17]	41.82%	64
BIVE	54.25%	2×64

‡ Double-precision feature vectors.

From Tables II and III, we observe that BIVE attains the highest CCR among the evaluated binary feature descriptors. A major disadvantage of SSH, LHS and IQT is that they require that double-precision feature vectors, in this case HOF feature vectors, be computed first. Thus, besides the computational complexity of the binary-mapping, it is important to consider the multiple calculations and storage requirements to extract and store the double-precision feature vectors. BIVE significantly outperforms 3D-FREAK, 3D-BRISK, 3D-ORB and 3D-BRIEF. This confirms the advantages of using wavelet-based patterns and relatively large regions to encode the temporal gradients of video volumes.

When tested on a BoF+SVM pipeline, BIVE outperforms BiTE by 10% and 7% on the KTH and UCF50 datasets, respectively, in terms of CCR. When combined, BiTE+BIVE outperform BIVE by 6% and 14% on the KTH and UCF50 datasets, respectively. BiTE+BIVE also outperform BiTE by 15% and 20% on the KTH and UCF50 datasets, respectively This proves that BIVE is more powerful than BiTE and when used together, they provide a stronger performance than when used separately.

It is important to mention that FREAK, BRISK and ORB require finding the best pairs of regions to be compared. The particular criterion used by these 2D feature descriptors to find these best pairs of regions, unfortunately, gives preference to those pairs that tend to produce binary strings with a mean $= 0.5$, i.e., binary strings where half of the bits are $1's$ and the other half are $0's$ [17, 18, 21]. According to Golomb's postulates, random binary strings tend to have the same number of $1's$ and $0's$ [63]. Therefore, this criterion gives



a) 2D FREAK patterns

b) 3D FREAK patterns

Fig. 6: Extension of the FREAK descriptor to 3D. **a)** Sample 2D FREAK patterns around a particular key point. Circular regions are progressively smaller as they are closer to the key point. b) 3D FREAK patterns defined in the spatio-temporal domain by using ellipsoids of three different sizes organized into three rings. The first ring (outermost layer) comprises the largest ellipsoids. The second and third rings comprise progressively smaller ellipsoids emulating the 2D FREAK patterns. The top row depicts the actual spatio-temporal regions to be compared, while the bottom row depicts the projected ellipsoids on the xyt planes for visualization purposes.

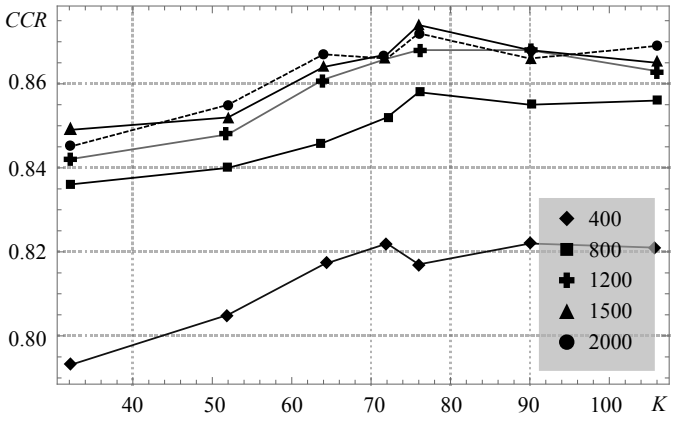


Fig. 7: CCR for the KTH dataset against K patterns for $\{400, \ldots, 2000\}$ centroids using a BoF+SVM pipeline.

preference to pairs of regions that tend to produce random binary strings, which defeats the purpose of creating highly descriptive feature vectors. In our experiments, we obtain more descriptive feature vectors for 3D-FREAK, 3D-BRISK and 3D-ORB by selecting those region pairs that produce strings whose mean are within the 2σ region, where σ is the standard deviation. It is important to emphazise that BIVE employs K patterns to define the pairs of regions to be compared and, therefore, does not require searching for the best pairs. In this work $K = 64$, which is a number of patterns that have been shown to provide the best trade-off between CCR and computational complexity. This is depicted in Fig. 7, where the CCR for the KTH dataset is plotted against K for a total of $\{400, 800, 1200, 1500, 2000\}$ centroids using a BoF+SVM pipeline and a 8/9 split [62]. Note that for $K > 64$ patterns, BIVE does not attain a significant increase in CCR values for any of the number of centroids evaluated.

B. Second set of experiments: STB complexity

This experiment compares the memory demands and computational time of the STB descriptor, with and without using the IV technique in BIVE, against several popular binary and non-binary feature descriptors. In this experiment, we use our 3D versions of FREAK and BRIEF. The results of this experiment are tabulated in Table IV, for a single video volume.

TABLE IV: Characteristics of various feature descriptors.

Feature descriptor	Bytes per feature	Computational time
3DSIFT [31]	18432	220 ms
MBH [34]	1728	100 ms
HOF [33, 60]	810	57 ms
HOG [33, 60]	640	43 ms
3D-FREAK †	4 – 128	(12 – 40) ms
3D-BRIEF †	8 – 128	(10 – 50) μs
STB (without IV) †	23	7.5 ms
STB (with IV) †	23	180 μs

† Binary.

From Table IV, we observe that using the IV technique indeed speeds up the encoding process by approximately **40**$\times$ compared to the case of not using it. We also observe that STB, when using the IV technique, is c.a. **1200**$\times$ faster than 3DSIFT, and **200**$\times$ faster than HOG. In terms of memory demands, feature vectors generated by STB are c.a. **830**$\times$ more compact than those generated by 3DSIFT, and **30**$\times$ more compact than those generated by HOG. These results confirm the advantages of the proposed STB descriptor in terms of computational times and memory demands, as well as the benefits of using the IV technique to reduce the number of computations. A detailed analysis of the complexity of the STB descriptor is provided in Appendix C.

To further evaluate and compare the STB complexity in terms of storage requirements for the extracted features, the

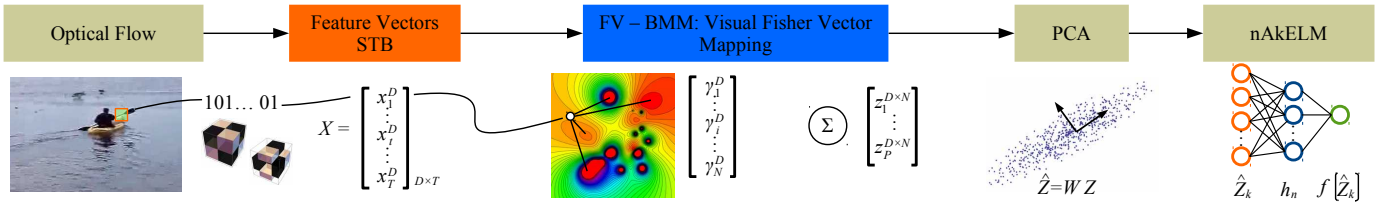


Fig. 8: Three-stage pipeline used to evaluate the proposed STB descriptor when paired with FVs. In the first stage, we extract STB feature vectors. The second stage projects the extracted binary feature vectors using FVs. In the last stage, the dimensions of the projected feature vectors are reduced using PCA; classification is done using an nAkELM.

hashing techniques SSH, LSH and IQT are applied to the IDT+MBH/HOG/HOF descriptor [34]. The compressed descriptors are tested on a BoF+SVM pipeline using the KTH dataset. The resulting CCRs are tabulated in Table V. As expected, applying a hashing algorithm reduces the amount of data needed to represent the feature descriptors, but may negatively affect the performance. Even after applying hashing techniques to IDT+MBH/HOG/HOF, our STB attains a very competitive CCR and has the lowest storage requirements.

TABLE V: CCR for the KTH dataset under a BoF+SVM pipeline using hashing techniques.

Descriptor	Storage	CCR
IDT+MBH/HOG/HOF [34]	5.85 GB	94.44%
IQT-IDT+MBH/HOG/HOF (64 bits)	5.9 GB	92.45%
SSH-IDT+MBH/HOG/HOF (64 bits)	5.9 GB	92.13%
LSH-IDT+MBH/HOG/HOF (64 bits)	5.9 GB	86.11%
STB	40.9 MB	93.05%

C. Third set of experiments: action recognition

We pair our STB with our FVs (as computed in Section IV) in the pipeline depicted in Fig. 8, where the FVs are dimensionally reduced using PCA, as this has been shown to reduce noisy feature projections over the BMM components [64–66]. We employ first order-features for the FVs. The pipeline in Fig. 8 employs for classification an Nigstrom Approximation Kernel Extreme Learning Machine (nAkELM) [67] trained with the dimensionality-reduced FVs. This particular machine can be trained faster and has been shown to be more accurate than SVMs [67]. We use a greedy tuning to set the number of PCA components and the sample size and constraint factor of nAkELM [67].

1) UCF50: We compare STB+FV using the pipeline in Fig. 8 against several state-of-the-art action recognition methods that have been tested on the UCF50 dataset. Specifically, these methods are the best-performing ones that use high-order representations and double-precision feature vectors [6, 33, 34], double-precision feature vectors with no high-order representations [61, 68], and binary feature vectors [47]. The results of these comparisons are reported in Table VI, which correspond to the average CCR over the splits.

From Table VI, we observe that STB+FV achieves a very competitive performance compared to the best performing non-binary methods, outperforming GIST3D and c3DSIFT by c.a **10%** and **15%**, respectively. It is worth noticing

TABLE VI: CCR of various action recognition methods on the UCF50 dataset.

Method	CCR
MIFS [6]	94.4%
IDT+MBH/HOF/HOG+FV [34]	91.2%
DT/MBH/HOF/HOG+SVM [33]	84.5%
GIST3D [68]	73.7%
c3DSIFT [61]	68.2%
MIPs [47]†	72.70%
STB+FV†	83.05%

† Binary.

that STB+FV is about **10%** more accurate than MIPs. Even though STB+FV is about **10%** less accurate than the methdos in [6, 34], it is important to mention that these methdos usually require a long time, in the order of days, just to extract the feature vectors for this dataset. STB+FV requires approximately **8.5 hours** to extract the feature vectors for the whole UCF50 dataset and only **3.3 GB** to store them, which is significantly smaller than the **846 GB** required by the methods in [6, 34].

2) UCF101: We compare STB+FV using the pipeline in Fig. 8 against several state-of-the-art action recognition methods, including methods based on CNNs. To the best of our knowledge, no other method that employs binary feature descriptors has been tested on this dataset. We evaluate 1) the CNN methods proposed in [12, 69–74], 2) the method proposed in [34], which uses high-order representations and double precision feature vectors, and 3) the method presented in [75], which is among the best-performing ones that use double-precision feature vectors with no high-order representations. The results of this evaluation, in terms of the average CCR over the splits, are reported in Table VII. These results include the number of parameters of each method and the required hardware for operation. The evaluated methods are tabulated into two sections: the first section lists those that require GPUs, while the second section lists those that can operate on CPUs. From Table VII, we observe that STB+FV achieves a very competitive performance compared to some of the CNN methods. For example, STB+FV is only **7%** less accurate than the CNN method in [73]. STB+FV outperforms the CNN method in [12], and the double-precision feature based method in [75], by **7%** and **27%**, respectively. Methods based on CNNs indeed attain the highest CCRs. This comes, however, at a high computational cost (see no.

TABLE VII: No. of parameters, hardware requirements and CCR of various methods for the UCF101 dataset.

Method	Parameters	Hardware	CCR
Methods that require GPUs			
I3D + PoTion [69]	25M+143M	65 GPUs	98.2%
Two-Streams I3D [70]	25M	32 GPUs	98%
DTPP [72]	12M+ 60.93M	3 GPUs	98%
RGB + OFF(RGB) + OFF(optical flow) + OFF(raw-OFF) [71]	44.2M	4 GPUs	96%
Two Stream-CNets [74]	25M	GPUs*	88.8%
EMV-CNets [73]	32.8M	1 GPU	79.3%
MultiRes-CNets [12]	40.3M	GPUs*	65.4%
Methods that require CPUs			
IDT/MBH/HOF/HOG+VLAN+FV [34]	663.5K	CPUs*	85.9%
H3D+HOF/HOG+SVM [75]	1.05M	1 CPU	43.9%
STB+FV†	356.3K	1 CPU	71.6%

† Binary.
* Arrays of GPUs or CPUs.

of parameters and hardware requirements). For example, I3D + PoTion (25M+143M parameters) requires two pre-trained CNNs as feature generators. One CNN uses 3D convolutions (I3D) and is pre-trained on a large scale dataset of human actions (Kinetics dataset). The second CNN estimates human joints and is pre-trained on another large dataset (COCO dataset) using a frame-by-frame analysis. The Two-Streams I3D (25M parameters) and DTPP (12M+60.93M parameters) methods also require pre-training on a large scale dataset of human actions (Kinetics dataset). In contrasts, STB+FV has only **356.3K** parameters for the UCF101 dataset.

3) Binary pipeline: To further verify the superiority of STB against other binary feature descriptors, we also use the pipeline depicted in Fig. 8 to evaluate the accuracy of the hashing techniques IQT and SSH when applied to IDT+MBH/HOG/HOF [34]. We also evaluate BDT, which is the previous version of our binary feature descriptor [23]. In order to have a fair comparison, all descriptors are evaluated in conjunction with our FVs. It is important to recall that IDT+MBH/HOG/HOF is one of the best methods in terms of accuracy, as shown in Table VI. Specifically, we employ these descriptors in lieu of STB (orange block of Fig. 8) with exactly the same spatio-temporal detector. For the case of BDT, we replace STB (orange block of Fig. 8) with BDT and use the same spatio-temporal detector. BDT follows the same approach as that of STB, but produces shorter feature vectors of $3L+K$ dimensions, where L is the number of displacement vectors and K the number of patterns, as defined before. For this evaluation, we compute the CCR for the UCF50 dataset. The results of this experiment are tabulated in Table VIII.

From Table VIII, we observe significant improvements of STB of nearly **3**% over SSH-IDT+MBH/HOG/HOF. STB also outperforms IQT-IDT+MBH/HOG/HOF by nearly **1**%. In terms of storage requirements, STB drastically reduces storage requirements by **240**×. Finally, STB requires a run time **5**× shorter than that required by SSH-IDT+MBH/HOG/HOF and IQT-IDT+MBH/HOG/HOF. Compared to BDT, STB achieves a CCR value **4**% higher. As expected, this improvement comes at the expense of longer run times and higher storage requirements. Let us recall that BDT employs only three bits to binarize the trajectories and does not encode separately the two motion components of the video volumes. Consequently, the resulting binary feature vectors are shorter, but as shown in Table VIII, less powerful than those generated by STB.

TABLE VIII: Performance of binary feature descriptors on the UCF50 dataset.

Method	Run Time	Storage	CCR
IQT-IDT+MBH/HOG/HOF+FV(64 bits) [48]	47.2 h	849.4 GB	82.12%
SSH-IDT+MBH/HOG/HOF+FV(64 bits) [44]	46.8 h	849.4 GB	80.56%
BDT+FV [23]	6.32h	2.43 GB	78.16 %
STB+FV	9.17h	3.29 GB	83.05%

D. Fourth set of experiments: gait recognition and animal behavior clustering

To demonstrate the generality of the STB descriptor, we also use the CASIA-B [76] and TIGdog [77] datasets for evaluation. For the CASIA-B dataset, we use the pipeline in Fig. 8. For the TIGdog dataset, we use the STB descriptor with FVs and $k-$ means, as the objective is to cluster similar animal behaviors.

For the CASIA-B dataset, the performance is evaluated in terms of the recognition rate ($[0, 1]$) for probe view angles $[18°, 36°, 54°, 72°]$ against the 90°gallery view angle. In other words, training is done using exclusively 90°data. The results of our STB descriptor and other methods are tabulated in Table IX. The STB descriptor outperforms the other evaluated methods for probe view angles [54°, 72°] by up to **20**%, despite of the fact that STB is not specifically designed to deal with appearance transformations. It is worth mentioning that the other compared methods implicitly compensate for the angle variations, and thus attain a better performance for the most challenging cases, i.e., 18°and 36°angles. STB does not compensate for such variations. Instead, STB uses the raw videos to extract features, which also implies that there is no need to detect human silhouettes to compute the gait energy images commonly used in gait recognition. STB, consequently, simplifies this aspect of the gait recognition task. Overall, results in Table IX show that the STB descriptor is suitable for gait recognition.

TABLE IX: Gait recognition rates of various methods on the CASIA-B dataset.

Method	18 °	36 °	54 °	72 °
DATER [78]	8%	18%	59%	96%
VTM [79]	30%	46%	63%	83%
MvDA [80]	27%	36%	64%	95%
STB+FV	6.9%	23.9%	84.9%	98.9%

For the TIGdog dataset, the performance is evaluated in terms of the purity and the Adjusted Rand Index (ARI). The purity is the number of centroids correctly clustered divided

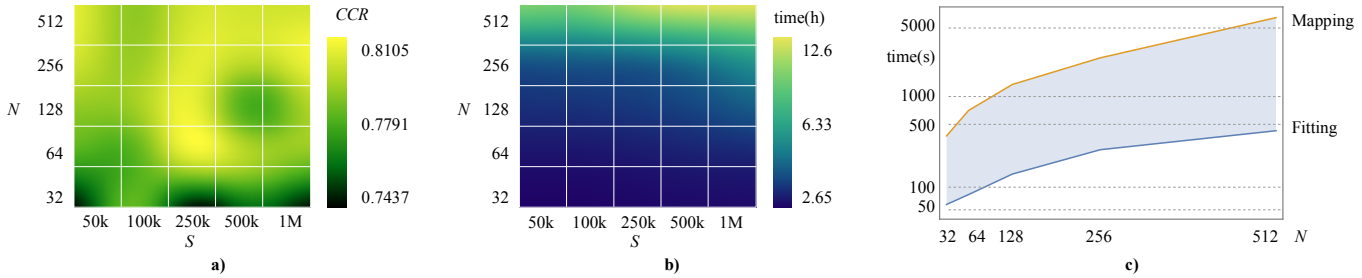


Fig. 9: FV parameter evaluation on split-1 of the UCF50 dataset. **a)** CCR for N BMM components fitting S STB feature vectors. **b)** Computational time of FV for N BMM components fitting S STB feature vectors. **c)** Computational time of the fitting and mapping processes of FV for N BMM components fitting $S = 100k$ STB feature vectors.

by the total number of features. The ARI is similar to the purity, but penalizes the assignment of two features with the same label to different clusters. The results for the TIGdog dataset are tabulated in Table X. These results include the requirements to store the features extracted by each evaluated method and the time taken to extract the features. STB achieves a very competitive performance compared to other methods that employ double-precision features. Moreover, STB requires **140**$\times$ less storage space than PoTs and IDT, and **30**$\times$ less storage than HOG. The time required to extract features is also reduced by nearly **6**$\times$ and **4**$\times$ compared to PoTs and IDT, and HOG, respectively.

TABLE X: Clustering performance of various methods on the TIGdog dataset.

Method	Purity	ARI	Storage	Feature extraction time
PoTs [77]	38	0.075	62.59 GB	5.12 h
IDT [34]	37	0.05	62.59 GB	5.12 h
HOG [60]	36	0.04	12.9 GB	3.48 h
STB+FV	36	0.065	433.3 MB	0.88 h

E. Fifth set of experiments: FV parameter evaluation

For this last set of experiments, we employ split-1 of the UCF50 dataset. We compute the CCR attained by the pipeline depicted in Fig. 8 and the computational time of our FVs when a different number of BMM components are used with a different number of STB feature vectors. These are the most influential parameters of the computations of FVs.

We first use N BMM components with X_S binary features to be fitted, where X_S is a subset of X with S features randomly selected. The results of this experiment are shown in Fig. 9.a and 9.b. We observe that the CCR is particularly high for $N > 64$ BMM components and $S > 250k$ STB feature vectors. As expected, computational times are also particularly high for $N > 64$ and $S > 250k$ (see Fig. 9.b). Experimentally, we find that $S = 250k$ STB feature vectors using $N = 256$ BMM components is a good tradeoff between CCR and computational times. For these particular set of values, the computational time of FV is about 2.2 hours (see Fig. 9.b). Finally, Fig. 9.c plots the computational time of the fitting and mapping processes of FV for N BMM components fitting $S = 100k$ STB feature vectors. From this Fig., we can observe that mapping is indeed the process that takes the longest and has an exponential behavior with respect to the number of BMM components.

F. Discussions about computational complexity

Although some CNN methods and some methods based on double-precision feature vectors can attain higher CCRs than those attained by STB+FV, their training and operational times can be considerably long. CNN methods usually require dedicated clusters or servers with arrays of GPUs [74], and may require months for training [12]. The same drawback is shared by methods that employ double-precision feature vectors. Even when using dedicated servers, they may require several days to process the extracted feature vectors [6, 34]. On the contrary, STB+FV takes only hours to process large datasets. For example, for the UCF101 dataset, it takes approximately **16 + 9 hours** (see implementation details in Appendix D). Moreover, storing double-precision feature vectors demands a high number of resources. For example, the double-precision feature vectors used in [33, 34] require around **1.7 TB** of storage for the UCF101 dataset, which is significantly larger than the **5 GB** storage requirements of STB. Table XI summarizes the computational demands of STB and FV as paired in the pipeline of Fig. 8 when tested on a single CPU (see Appendix D). From this table, we can observe that STB and FV require significantly low computational resources for both UCF50 and UCF101.

TABLE XI: STB and FV computational demands

Dataset	STB storage demands	FV run time
UCF101	5.21GB	8.78h
UCF50	3.29GB	6.35h

Another important aspect to consider is the number of parameters of many state-of-the-art CNNs, which can be in the order of millions (see Table VII as an example). CNNs can also require billions of computations to classify a single input instance [81, 82]. Moreover, CNNs produce several intermediate feature maps, which must be stored in memory. CNN compression methods can indeed reduce the memory size required to store parameters and intermediate feature maps, as well as to reduce the overall number of computations. This is usually done by pruning connections, quantizing

connection weights, and applying Huffman encoding to the remaining quantized connection weights [82, 83]. However, even if compression is applied, the original CNN must be first fully trained, which still demands memory to store intermediate feature maps and computational power to perform all operations. Moreover, pruning and weight quantization may affect the overall accuracy of the CNN [81].

Although GPU/TPUs can accelerate computing for methods based on CNNs, such computational power may not be available in embedded applications, low-cost devices, or for large-scale operation, such as in the case of video surveillance using data from several cameras. Let us take the VGG-16 CNN as an example, which is widely used for image classification [84]. This CNN has over 138 million network parameters, which may require over 276 MB of memory space if a 16-bit number representation is used. Additionally, during operation, VGG-16 CNN produces several feature maps that must be stored internally in memory. Specifically, the largest feature map of the VGG-16 CNN is more than 6 MB. Since the VGG-16 CNN comprises 21 layers, the amount of data to be moved internally in memory may easily reach 60 MB. In terms of the number of operations, the VGG-16 CNN may need over 15G of multiply-accumulate (MAC) operations to classify one input image. Consequently, the deployment of CNNs in embedded applications and low-cost devices remains very challenging, especially when latency and throughput is a main concern.

The number of operations and memory requirements needed to extract and store the proposed STB descriptor are much lower than those of many CNN-based methods. Specifically, the number of operations and memory requirements of STB are, on average, linear and depend on n, the number of pixel locations of the spatio-temporal region, and the L, the number of displacements comprising each trajectory (see Appendix C for a detailed explanation). Moreover, the fact that the STB descriptor produces binary data allows performing subsequent computations using binary operators like AND and XOR, which further guarantees a low computational complexity. Combining hand-crafted binary feature descriptors, like STB, with low-complexity classifiers or low-complexity neural networks (e.g., the nAkELM) is indeed an attractive solution for embedded applications and low-cost devices.

VI. Conclusions

In this work, we presented Spatio-Temporal Binarization (STB), a low-complexity and compact binary feature descriptor. STB encodes motion information from two sources, namely, optical flow and temporal gradients. Compared to our previous work, the performance of STB is improved in two aspects. First, it allows to distinguish motion information from motion camera and other small random movements. This is achieved by using longer binary codes to encode the constituent displacements of trajectories obtained from optical flow. Second, it allows to better describe the motion captured by temporal gradients by independently encoding the horizontal and vertical motion components. Furthermore, STB's complexity is further reduced by employing a technique based on Integral Videos. Extensive evaluations for action recognition, gait recognition and animal behavior clustering using the KTH, UCF50, UCF101, CASIA-B, and TIGdog datasets confirmed the advantages of STB in terms of processing times and storage requirements. These evaluations showed that when paired with the proposed Fisher Vectors for binary data, STB attains a very competitive accuracy, outperforming several state-of-the-art non-binary and binary features descriptors, including methods based on CNNs.

Appendix A
Fisher Score

By substituting 13a into Eq. 16b, we have:

$$\mathcal{L}\left(X|\theta\right) = \log\left(\prod_{1\leqslant t\leqslant T} p\left(x_t|\theta\right)\right), \tag{25a}$$

$$\because \log(\prod_{k} a_k) = \sum_{k}(\log(a_k)), \tag{25b}$$

$$\mathcal{L}\left(X|\theta\right) = \sum_{1\leqslant t\leqslant T} \log\left(p\left(x_t|\theta\right)\right). \tag{25c}$$

The Fisher Score, $\mathcal{G}_\theta^X$, can then be expressed as:

$$\mathcal{G}_\theta^X = \frac{1}{T}\nabla_\theta \sum_{1\leqslant t\leqslant T} \log\left(p\left(x_t|\theta\right)\right), \tag{26a}$$

$$= \frac{1}{T} \sum_{1\leqslant t\leqslant T} \partial_\theta \log\left(p\left(x_t|\theta\right)\right), \tag{26b}$$

$$= \frac{1}{T} \sum_{1\leqslant t\leqslant T} \frac{1}{p\left(x_t|\theta\right)} \partial_\theta \left(p\left(x_t|\theta\right)\right). \tag{26c}$$

As [51] suggests, we derive the BMM μ_{id} parameter over the distribution and independence of w_i from θ and not by only examining the two possible outcomes of x_t, as done in [55]. We then have:

$$\partial_\theta p(x_t|\theta) = \sum_{1\leqslant i\leqslant N} \partial_{\mu_{id}}\left(w_i p_i\left(x_t|\theta\right)\right), \tag{27a}$$

$$= \sum_{1\leqslant i\leqslant N} w_i \partial_{\mu_{id}}\left(p_i\left(x_t|\theta\right)\right). \tag{27b}$$

We then estimate $\partial_{\mu_{id}}\left(p_i\left(x_t|\theta\right)\right)$ from Eq. 13c. At this point, we fully expand the partial derivative and avoid simplifying the term only by its two possible values, as done in [55]:

$$\partial_{\mu_{id}} p_i(x_t|\theta) = \prod_{1\leqslant d\leqslant D} \partial_{\mu_{id}}\left(\mu_{id}^{x_{td}}\left(1-\mu_{id}\right)^{1-x_{td}}\right). \tag{28}$$

Simplifying the notation; i.e., $x_{td} \rightarrow x, \mu_{id} \rightarrow \mu$, gives us:

$$\prod_{1\leqslant d\leqslant D} \partial_\mu \left(\mu^x(1-\mu)^{1-x}\right), \tag{29a}$$

$$\prod_{1\leqslant d\leqslant D} x(1-\mu)^{1-x}\mu^{x-1} - (1-x)(1-\mu)^{-x}\mu^x, \tag{29b}$$

$$\prod_{1\leqslant d\leqslant D} (1-\mu)^{-x}(x-\mu)\mu^{x-1}, \tag{29c}$$

$$\prod_{1\leqslant d\leqslant D} \mu^x(1-\mu)^{1-x}\frac{x-\mu}{\mu(1-\mu)}, \tag{29d}$$

$$\underbrace{\prod_{1\leqslant d\leqslant D} \mu_{id}^{x_{td}}(1-\mu_{id})^{1-x_{td}}}_{p_i(x_{td}|\theta)} \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}, \tag{29e}$$

$$\therefore \partial_{\mu_{id}} p_i(x_t|\theta) = p_i(x_{td}|\theta) \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}. \tag{29f}$$

After substituting 29f in 27b, we have:

$$\partial_\theta p(x_t|\theta) = \sum_{1\leqslant i\leqslant N} w_i p_i(x_{td}|\theta) \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}. \tag{30}$$

After substituting Eq. 14 in Eq. 26c, we finally have:

$$\frac{1}{T}\sum_{1\leqslant t\leqslant T} \underbrace{\frac{1}{p\left(x_t|\theta\right)} \sum_{1\leqslant i\leqslant N} w_i p_i(x_{td}|\theta)}_{\gamma_t(i)} \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}, \tag{31a}$$

$$\therefore \mathcal{G}^X_{\mu_{id}} = \frac{1}{T}\sum_{1\leqslant t\leqslant T} \gamma_t(i) \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}. \tag{31b}$$

APPENDIX B
FISHER INFORMATION MATRIX

It can be shown that the first moment of the Fisher Score, i.e., its expected value, is 0. Thus the second moment, which corresponds to the Fisher Information, is given as follows:

$$F_{\mu_{id}} = E\left[\left(\partial_\theta \mathcal{L}\left(X|\theta\right)\right)^2 |\theta\right], \tag{32a}$$

$$= \int_{x_t} p(x_t|\theta)\left(\partial_\theta \log p\left(X|\theta\right)\right)^2 dx_t. \tag{32b}$$

After simplifying the integration required by Eq. 32a, we have:

$$p(x_t|\theta)\left(\partial_\theta \log p\left(X|\theta\right)\right)^2, \tag{33a}$$

$$p(x_t|\theta)\left(\sum_{1\leqslant t\leqslant T} \gamma_t(i) \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}\right)^2. \tag{33b}$$

Assuming that the derivative of the posterior probability of $\gamma_t(i)$ is sharply peaked, we can simply write $\gamma_t(i)^2 \approx \gamma_t(i), \forall i$. Unlike the work in [55], we derive the information matrix by assuming two separate integration terms and merging them only when calculating it over T number of features. Then we have for x_t:

$$F_{\mu_{id}} = \int_{x_t=0} \ldots dx_t + \int_{x_t=1} \ldots dx_t. \tag{34}$$

The sharply peaked derivative for the integrals is simplified as:

$$\left.\frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}\right|_{x_t=0} = -\frac{1}{1-\mu_{id}}, \tag{35a}$$

$$\left.\frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}\right|_{x_t=1} = \frac{1}{\mu_{id}}. \tag{35b}$$

Therefore, we have two separate integration terms:

$$F_{\mu_{id}} = \int_{x_t=0} p(x_t|\theta) \sum_{1\leqslant t\leqslant T} \frac{\gamma_t(i)^2}{\left(1-\mu_{id}\right)^2} dx_t + \int_{x_t=1} p(x_t|\theta) \sum_{1\leqslant t\leqslant T} \frac{\gamma_t(i)^2}{\mu_{id}^2} dx_t, \tag{36}$$

$$= \sum_{1\leqslant t\leqslant T} \int_{x_t=0} p(x_t|\theta)\gamma_t(i) \frac{1}{\left(1-\mu_{id}\right)^2} dx_t + \sum_{1\leqslant t\leqslant T} \int_{x_t=1} p(x_t|\theta)\gamma_t(i) \frac{1}{\mu_{id}^2} dx_t. \tag{37}$$

Considering that $p(x_t|\theta)\gamma_t(i) = w_i p_i(x_t|\theta)$, Eq. 37 becomes:

$$F_{\mu_{id}} = \sum_{1\leqslant t\leqslant T} \int_{x_t=0} \frac{w_i p_i(x_t|\theta)}{\left(1-\mu_{id}\right)^2} dx_t + \sum_{1\leqslant t\leqslant T} \int_{x_t=1} \frac{w_i p_i(x_t|\theta)}{\mu_{id}^2} dx_t. \tag{38}$$

After evaluating the integral, we have:

$$F_{\mu_{id}} = \sum_{1\leqslant t\leqslant T} \frac{w_i \mu_{id}}{\mu_{id}^2} + \sum_{1\leqslant t\leqslant T} \frac{w_i(1-\mu_{id})}{\left(1-\mu_{id}\right)^2}, \tag{39}$$

which finally leads to:

$$F_{\mu_{id}} = T w_i \left(\frac{1}{\mu_{id}} + \frac{1}{1-\mu_{id}}\right), \tag{40}$$

or equivalently expressed using a single quotient:

$$F_{\mu_{id}} = \frac{T w_i}{\mu_{id} - \mu_{id}^2}. \tag{41}$$

Expression in Eq. 41 approximates the Fisher information matrix for a GMM [51]. Our information matrix is significantly simpler than that in [55], requiring one single term evaluation.

APPENDIX C
COMPLEXITY OF THE STB DESCRIPTOR

BiTE complexity: Let us recall that BiTE is computed by concatenating L binarized displacement vectors using Eq. 3, where each displacement vector, $\hat{p}_t$, is mapped into one of six 4-bit binary codes, using Eq. 2. Because $B(\Delta\hat{p}_t)$ in Eq. 2 can be calculated in linear time, the sum in Eq. 3 has a complexity of $O(n^2)$. This complexity represents the worst case scenario, i.e., when it is necessary to calculate the L binary codes using the $\arg\min$ expression (if $|\Delta\hat{p}_t| \in (\epsilon, 1-\epsilon)$) against Q bins, which results in $Q \times L$ operations. For the best case scenario, either $|\Delta\hat{p}_t| \geq 1-\epsilon$ or $|\Delta\hat{p}_t| \leq \epsilon$, the complexity is then $O(n)$.

BiTE memory requirements: To compute Eq. 3, the trajectory and the Q bins must be stored. Each trajectory is made of 3 displacements, (x, y, t), thus $3n$ variables are required. The bins are stored as polar angles, thus requiring $2Q$ constants. Finally, we need to store each binarized trajectory, which requires $4n$ variables. The total memory requirements are then $7n + 2Q$ per BiTE feature.

BIVE complexity: To compute BIVE, the horizontal and vertical motion components of volume v_i must be computed first using Eq. 4a and 4b. Since two gradients are applied to compute each component, Eq. 4a and 4b require each $2n$ operations for the n pixels locations of v_i. For each component, determining the coordinates of the centroid, c, requires multiplying the value of every pixel location by its location, which can be computed in n operations, according to Eq. 5. Computing the angle θ requires only one operation (see Eq. 6). The flip operation requires to re-allocate n pixels locations, thus n operations are required (see Eq. 7). Calculating the Integral Video, as described in Algorithm 1, can be computed with n accumulations for n pixel locations (see Eq. 10). Finally, for the kth pattern, the expression in Eq. 11 must be evaluated.

For the $K = 64$ patterns used in this work, 1056 operations are required for each video component, with an average of 150 OPB (or pattern). If the flip operation is required, which represents the worst case scenario, the total number of operations for the n pixel locations of video volume v_i is $2(4n + 1056 + 1)$. For the best case scenario, i.e., when no flip operation is required, the total number of operations is $2(3n + 1056 + 1)$. Thus, the complexity of BIVE is $O(n)$ for n pixel locations.

BIVE memory requirements: Given the n pixel locations of video volume v_i, the memory requirements are $2n$ for the two components. When evaluating the Integral Video, the same memory spaces assigned to the video volume v_i are used in the recursive operation, thus no additional memory is required. The total memory requirements of BIVE are then $2n + 2K$ for K patterns.

APPENDIX D
IMPLEMENTATION DETAILS

We implement STB+FV (pipeline in Fig. 8) in MATLAB and perform evaluations on a single 2.7GHz CPU with 16GB RAM memory. We use the homogeneous patch detector proposed in [34] as implemented by the authors. Video volumes used by STB are of size $n_x = 32, n_y = 32, n_t = 15$. For BiTE, we use $\epsilon = 0.1$ to encode the displacement vectors. We use the nAkELM implementation provided by the authors [67]. For evaluations on the KTH dataset, we use the STIP detector as implemented by the authors of [37] and the χ^2-SVM as implemented by the corresponding authors [1]. The size of the video volumes used in Section V-A are $n_x = n_y = 9\sigma$ and $n_t = 9\tau$, where σ and τ are the scales at which the corresponding STIP is detected.

[1]https://goo.gl/chhfq0

REFERENCES

[1] Cisco, "Cisco visual networking index: Forecast and methodology, 2014-2019 white paper," https://goo.gl/xoBrTA.

[2] O. Popoola and K. Wang, "Video-based abnormal human behavior recognition;a review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,2012*, vol. 42, no. 6, pp. 865–878, Nov 2012.

[3] H. Nam and C. D. Yoo, "Content adaptive video summarization using spatio-temporal features," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 4003–4007.

[4] J. Wang, W. Wang, and W. Gao, "Multi-scale deep alternative neural network for large-scale video classification," *IEEE Transactions on Multimedia*, pp. 1–1, 2018.

[5] J. Zheng, X. Cao, B. Zhang, X. Zhen, and X. Su, "Deep ensemble machine for video classification," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2018.

[6] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj, "Beyond gaussian pyramid: Multi-skip feature stacking for action recognition," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, June 2015, pp. 204–212.

[7] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, June 2015, pp. 4305–4314.

[8] H. Dee and S. Velastin, "How close are we to solving the problem of automated visual surveillance?" *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 329–343, 2008.

[9] N. Haering, P. Venetianer, and A. Lipton, "The evolution of video surveillance: an overview," *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 279–290, 2008.

[10] Y. Cong, J. Yuan, and J. Liu, "Abnormal event detection in crowded scenes using sparse representation," *Pattern Recognition*, vol. 46, no. 7, pp. 1851 – 1864, 2013.

[11] R. Leyva, V. Sanchez, and C. T. Li, "Video anomaly detection with compact feature sets for online performance," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3463–3478, July 2017.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1725–1732.

[13] M. Bilal, A. Khan, M. U. K. Khan, and C. M. Kyung, "A low-complexity pedestrian detection framework for smart video surveillance systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 10, pp. 2260–2273, Oct 2017.

[14] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 971–987, Jul 2002.

[15] S. Saha and V. Demoulin, "Aloha: An efficient binary descriptor based on haar features," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*, Sept 2012, pp. 2345–2348.

[16] Y. Ma and P. Cisar, "Event detection using local binary pattern based dynamic textures," in *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, June 2009, pp. 38–44.

[17] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 2548–2555.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 2564–2571.

[19] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision – ECCV 2010*, ser. Lecture Notes in Computer Science,

K. Daniilidis, P. Maragos, and N. Paragios, Eds. Springer Berlin Heidelberg, 2010, vol. 6314, pp. 778–792.
[20] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 5, pp. 815–830, May 2010.
[21] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 510–517.
[22] R. Leyva, V. Sanchez, and C. T. Li, "A fast binary pair-based video descriptor for action recognition," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sept 2016, pp. 4185–4189.
[23] R. Leyva, V. Sanchez, and T. L. Chang, "Fast binary-based video descriptors for action recognition," in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Nov 2016, pp. 1–8.
[24] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 3384–3391.
[25] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, Oct 2005, pp. 65–72.
[26] G. Willems, T. Tuytelaars, and L. Van Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector," in *Computer Vision – ECCV 2008*, ser. Lecture Notes in Computer Science, D. Forsyth, P. Torr, and A. Zisserman, Eds. Springer Berlin Heidelberg, 2008, vol. 5303, pp. 650–663.
[27] L. Shao and R. Gao, "A wavelet based local descriptor for human action recognition." in *BMVC*, 2010, pp. 1–10.
[28] L. Shao, R. Gao, Y. Liu, and H. Zhang, "Transform based spatio-temporal descriptors for human action recognition," *Neurocomputing*, vol. 74, no. 6, pp. 962 – 973, 2011.
[29] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.
[30] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Computer Vision – ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Springer Berlin Heidelberg, 2006, vol. 3952, pp. 428–441.
[31] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *ACM Multimedia Conference*, 2007.
[32] A. Klaser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *BMVC 2008-19th British Machine Vision Conference*, 2008, p. 275.
[33] H. Wang, A. Klaser, A.ser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, 2013.
[34] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *2013 IEEE International Conference on Computer Vision*, Dec 2013, pp. 3551–3558.
[35] X. Peng, C. Zou, Y. Qiao, and Q. Peng, *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. Cham: Springer International Publishing, 2014, ch. Action Recognition with Stacked Fisher Vectors, pp. 581–595.
[36] R. Maani, S. Kalra, and Y. H. Yang, "Robust volumetric texture classification of magnetic resonance images of the brain using local frequency descriptor," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4625–4636, Oct 2014.
[37] I. Laptev and T. Lindeberg, "Space-time interest points," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct 2003, pp. 432–439 vol.1.
[38] P. Matikainen, M. Hebert, and R. Sukthankar, "Trajectons: Action recognition through the motion analysis of tracked features," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, Sept 2009, pp. 514–521.
[39] R. Messing, C. Pal, and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," in *Computer Vision, 2009 IEEE 12th International Conference on*, Sept 2009, pp. 104–111.
[40] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li, "Hierarchical spatio-temporal context modeling for action recognition," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, June 2009, pp. 2004–2011.
[41] J. Sun, Y. Mu, S. Yan, and L.-F. Cheong, "Activity recognition using dense long-duration trajectories," in *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, July 2010, pp. 322–327.
[42] B. Chakraborty, M. B. Holte, T. B. Moeslund, and J. Gonzàlez, "Selective spatio-temporal interest points," *Computer Vision and Image Understanding*, vol. 116, no. 3, pp. 396 – 410, 2012, special issue on Semantic Understanding of Human Behaviors in Image Sequences.
[43] L. Yeffet and L. Wolf, "Local trinary patterns for human action recognition," in *Computer Vision, 2009 IEEE 12th International Conference on*, Sept 2009, pp. 492–497.
[44] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1092–1104, June 2012.
[45] ——, "Kernelized locality-sensitive hashing for scalable image search," in *2009 IEEE 12th International Conference on Computer Vision*, Sept 2009, pp. 2130–2137.
[46] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1509–1517.
[47] O. Kliper-Gross, Y. Gurovich, T. Hassner, and L. Wolf, *Motion Interchange Patterns for Action Recognition in Unconstrained Videos*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 256–269.
[48] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, Dec 2013.
[49] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid, "Evaluation of gist descriptors for web-scale image search," in *Proceedings of the ACM International Conference on Image and Video Retrieval*. ACM, 2009, p. 19.
[50] C. Whiten, R. Laganiere, and G.-A. Bilodeau, "Efficient action recognition with mofreak," in *Computer and Robot Vision (CRV), 2013 International Conference on*. IEEE, 2013, pp. 319–325.
[51] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, pp. 1–8.
[52] F. Perronnin and D. Larlus, "Fisher vectors meet neural networks: A hybrid classification architecture," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3743–3752.
[53] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International Journal of Computer Vision*, vol. 105, no. 3, pp. 222–245, 2013.
[54] H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, Sept 2012.
[55] Y. Uchida and S. Sakazawa, "Image retrieval with fisher vectors

of binary features," in *2013 2nd IAPR Asian Conference on Pattern Recognition*, Nov 2013, pp. 23–28.

[56] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291 – 307, 1999.

[57] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. I–511–I–518 vol.1.

[58] A. Juan and E. Vidal, "Bernoulli mixture models for binary images," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3, Aug 2004, pp. 367–370 Vol.3.

[59] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *In Advances in Neural Information Processing Systems 11*. MIT Press, 1998, pp. 487–493.

[60] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–8.

[61] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of web videos," *Machine Vision and Applications*, vol. 24, no. 5, pp. 971–981, 2013.

[62] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3, Aug 2004, pp. 32–36 Vol.3.

[63] G. Everest, A. J. Van Der Poorten, I. E. Shparlinski, T. Ward *et al.*, *Recurrence sequences*. American Mathematical Society Providence, RI, 2003, vol. 104.

[64] V. Chandrasekhar, J. Lin, O. Morère, H. Goh, and A. Veillard, "A practical guide to cnns and fisher vectors for image instance retrieval," *Signal Processing*, vol. 128, pp. 426–439, 2016.

[65] V. Kantorov and I. Laptev, "Efficient feature extraction, encoding, and classification for action recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 2593–2600.

[66] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Computer Vision and Image Understanding*, vol. 150, pp. 109 – 125, 2016.

[67] A. Iosifidis and M. Gabbouj, "On the kernel extreme learning machine speedup," *Pattern Recognition Letters*, vol. 68, Part 1, pp. 205 – 210, 2015.

[68] B. Solmaz, S. M. Assari, and M. Shah, "Classifying web videos using a global video descriptor," *Machine Vision and Applications*, vol. 24, no. 7, pp. 1473–1485, 2013.

[69] V. Choutas, P. Weinzaepfel, J. Revaud, and C. Schmid, "Potion: Pose motion representation for action recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 7024–7033.

[70] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 4724–4733.

[71] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang, "Optical flow guided feature: A fast and robust motion representation for video action recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 1390–1399.

[72] J. Zhu, Z. Zhu, and W. Zou, "End-to-end video-level representation learning for action recognition," in *2018 24th International Conference on Pattern Recognition (ICPR)*, Aug 2018, pp. 645–650.

[73] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with enhanced motion vector cnns," *arXiv preprint arXiv:1604.07669*, 2016.

[74] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 568–576.

[75] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[76] S. Zheng, J. Zhang, K. Huang, R. He, and T. Tan, "Robust view transformation model for gait recognition," in *2011 18th IEEE International Conference on Image Processing*. IEEE, 2011, pp. 2073–2076.

[77] L. Del Pero, S. Ricco, R. Sukthankar, and V. Ferrari, "Articulated motion discovery using pairs of trajectories," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 2151–2160.

[78] N. Liu and Y. Tan, "View invariant gait recognition," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2010, pp. 1410–1413.

[79] Y. Makihara, R. Sagawa, Y. Mukaigawa, T. Echigo, and Y. Yagi, "Gait recognition using a view transformation model in the frequency domain," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 151–163.

[80] A. Mansur, Y. Makihara, D. Muramatsu, and Y. Yagi, "Cross-view gait recognition using view-dependent discriminative analysis," in *IEEE International Joint Conference on Biometrics*, Sep. 2014, pp. 1–8.

[81] R. Struharik, B. Vukobratović, A. Erdeljan, and D. Rakanović, "Conna–compressed cnn hardware accelerator," in *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2018, pp. 365–372.

[82] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[83] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, "Quantized cnn: a unified approach to accelerate and compress convolutional networks," *IEEE transactions on neural networks and learning systems*, no. 99, pp. 1–14, 2017.

[84] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.